AI Tutor Project Deconstruction (v1.0.0)

Overview: Deconstruction of the AI Tutor project (v1.0.0) to its core, explaining how all components work together, excluding CrewAI complexity.

🏗️ 1. AI Tutor v1.0.0 - Complete Core Deconstruction

📁 1.1. Project Structure & File Responsibilities

- **Core Project Structure:**

```
Agentic-AI-Tutor/
├── 🚀 app.py            # Main UI application (Gradio)
├── ⚙️ setup.py           # One-time setup (PDF → Vector Index)
├── 📋 requirements.txt   # Python dependencies
├── 🔑 .env              # API keys (user creates)
├── 📁 data/
│   ├── syllabi/        # User's PDF files
│   ├── vector_store/   # FAISS index (auto-generated)
│   └── cache/          # Temporary files
├── 📁 src/
│   ├── agents/
│   │   └── tutor_agent.py  # Core AI logic (RAG + API calls)
│   └── utils/
│       └── euriai_embeddings.py # Custom embedding wrapper
└── 📚 Documentation files...
```

⚙️ 2. Setup Phase - How PDFs Become Searchable

🔄 Setup Phase Workflow:

- **Step 1: Environment Check**

```
# setup.py checks:
✅ API key exists in .env file
✅ data/syllabi/ directory exists
✅ PDF files are present
```

- **Step 2: PDF Discovery & Parsing**

```
# For each PDF file like "CBSE_10th_Science.pdf":
filename = "CBSE_10th_Science.pdf"
parts = filename.split('_')  # ["CBSE", "10th", "Science"]
board, grade, subject = "CBSE", "10th", "Science"
```

```
# Parse PDF content:
loader = PyPDFLoader(file_path)                # Load PDF
documents = loader.load_and_split(text_splitter)   # Split into 1000-char chunks

# Add metadata to each chunk:
for doc in documents:
   doc.metadata = {
      'board': 'CBSE',
      'grade': '10th',
      'subject': 'Science',
      'source': 'data/syllabi/CBSE_10th_Science.pdf',
      'page': 15  # Page number from PDF
   }
```

- **Step 3: Text Embedding & Vector Creation**

```
# Convert text to numbers (vectors):
embedding_function = EuriaiEmbeddings()  # Our custom wrapper

# For each text chunk:
text = "Light travels in straight lines. This is evident from shadows..."
vector = embedding_function.embed_documents([text])
# Result: [0.123, -0.456, 0.789, ..., 0.234]  (1536 numbers)

# Create searchable index:
faiss_index = FAISS.from_documents(all_documents, embedding_function)
faiss_index.save_local("data/vector_store/faiss_index")
```

- **Output**: A searchable database of syllabus content!

🚀 3. Runtime Phase - How the App Works

- **Note:** For v1.0.0 (Basic Version), the focus is on the basic tutor.

🏁 App Startup (app.py lines 11-35)

```
# When you run "python app.py":
tutor_agent = TutorAgent()            # Initialize AI tutor
basic_ready = tutor_agent.retriever is not None  # Check if vector store loaded

# This tells us if setup.py worked:
print(f"🤖 Basic Tutor: {'✅' if basic_ready else '❌'}")
```

🖥️ User Interface Creation (Gradio)

- **For v1.0.0, the UI has:**

```
# Input Controls:
grade_dropdown = ["1st", "2nd", ..., "10th"]      # Student's grade
board_dropdown = ["CBSE", "ICSE", "Karnataka"]     # School system
subject_dropdown = ["Science", "Math", "Social"]   # Subject to study
age_dropdown = ["5", "6", ..., "15"]               # Student's age

# Two Main Features:
tab1 = "📅 My Learning Plan"   # Generate roadmap
tab2 = "💬 Ask Questions"      # Chat with AI
```

🎯 Core Functions (v1.0.0 Basic Version)

- **When user clicks "Create My Learning Plan":**

```
def generate_roadmap_interface(grade, board, subject):
  if not basic_ready:
    return "⚠️ Setup needed - run python setup.py first"

  # Call the core AI logic:
  roadmap = tutor_agent.generate_roadmap(grade, board, subject)
  return roadmap
```

- **When user types a question and hits "Send":**

```
def chat_with_tutor(message, history, grade, board, subject, age):
  if not basic_ready:
    bot_response = "🤖 Hi! I need to be set up first. Ask a grown-up to help!"
  else:
    # Call the core AI chat logic:
    bot_response = tutor_agent.chat_with_kid(message, grade, board, subject)

  # Add to chat history and return:
  history.append([message, bot_response])
  return "", history
```

🔄 4. Complete Data Flow - PDF to Final Response

📊 **Visual Data Flow:**

```
User PDF → Setup Process → Runtime Query → AI Response
   ↓         ↓           ↓           ↓
[CBSE_10th_  [Text Chunks]  [Vector      [Generated
 Science.pdf]  + Metadata     Search]      Roadmap]
```

🔍 **Detailed Flow:**

- **Phase A: Setup (One-Time)**
  1. **User Action:** Adds "CBSE_10th_Science.pdf" to data/syllabi/
  2. **User Action:** Runs "python setup.py"
  3. **setup.py Process:**
     - Scans data/syllabi/ for PDFs
     - Parses filename: "CBSE_10th_Science.pdf" → board="CBSE", grade="10th", subject="Science"
     - Extracts text: "Light travels in straight lines..."
     - Splits into chunks: 1000 characters each, 100 overlap
     - Adds metadata: {'board': 'CBSE', 'grade': '10th', 'subject': 'Science'}
     - Converts to vectors: [0.123, -0.456, 0.789, ...] via Euriai API
     - Creates FAISS index: Fast similarity search database
     - Saves to: data/vector_store/faiss_index/
  4. **Result:** Searchable database of syllabus content ready!
- **Phase B: Runtime (Every User Interaction)**
  1. **User Action:** Selects "CBSE", "10th", "Science" → Clicks "Generate Roadmap"
  2. **app.py Process:**
     - Calls: tutor_agent.generate_roadmap("10th", "CBSE", "Science")
     - Shows loading message: "🧠 Creating your learning plan..."
  3. **tutor_agent.py Process:**
     - Creates search query: "CBSE 10th Science syllabus topics"
     - Converts query to vector: [0.234, -0.567, 0.890, ...] via EuriaiEmbeddings
     - Searches FAISS index: Finds top 10 similar document chunks
     - Filters by metadata: Only chunks with board="CBSE", grade="10th", subject="Science"
     - Combines context: "Light travels in straight lines... Laws of reflection..."
     - Builds prompt: "Create weekly plan for 10th Science (CBSE)... Content: [context]"
     - Calls Euriai API: Sends prompt to gpt-4.1-nano

        - Returns: AI-generated roadmap in markdown format
4. **app.py Display:**
    - Receives: "# 🔬 Science Learning Roadmap..."
    - Shows in UI: Formatted markdown roadmap

🔢 **Example with Real Data:**

- **Input Data:**
    - **PDF Content (page 15):** "Light travels in straight lines. This is evident from the formation of shadows when light encounters an opaque object."
    - **Chunk Created:**
      ```
      {
        "page_content": "Light travels in straight lines. This is evident from the
      formation of shadows...",
        "metadata": {
          "board": "CBSE",
          "grade": "10th",
          "subject": "Science",
          "source": "data/syllabi/CBSE_10th_Science.pdf",
          "page": 15
        }
      }
      ```
    - **Vector:** [0.123, -0.456, 0.789, 0.234, -0.567, 0.890, ...] (1536 numbers)
- **User Query:**
    - **User selects:** grade="10th", board="CBSE", subject="Science"
    - **Search query:** "CBSE 10th Science syllabus topics"
    - **Query vector:** [0.234, -0.567, 0.890, ...]
- **Vector Search:**
    - FAISS finds similar chunks based on cosine similarity:
        - Similarity 0.89: "Light travels in straight lines..."
        - Similarity 0.85: "Laws of reflection state that..."
        - Similarity 0.82: "Refraction occurs when light..."
- **Prompt Created:**
  ```
  Create a weekly study plan for a 10th student studying Science (CBSE board).

  IMPORTANT: The content below is from the official CBSE 10th Science syllabus.
  Create the plan ONLY based on this content.

  Syllabus Content:
  ```

Light travels in straight lines. This is evident from the formation of shadows when light encounters an opaque object.

Laws of reflection state that the angle of incidence equals the angle of reflection. The incident ray, reflected ray, and normal all lie in the same plane.

Refraction occurs when light passes from one medium to another with different optical densities...

Make it friendly and organized with weeks/months. Use markdown formatting.

- **AI Response:**
  # 🔬 Science Learning Roadmap (10th Grade, CBSE)

  ## Week 1: Light - Reflection and Refraction
  - Understanding how light travels in straight lines
  - Shadow formation experiments with flashlight
  - Laws of reflection using mirrors

  ## Week 2: Human Eye and Vision
  - Structure of the human eye
  - How we see through refraction
  - Common vision problems and corrections

🔌 5. Euriai API Integrations - Both Endpoints

🎯 Two Euriai API Endpoints Used:

- **Endpoint 1: /embeddings (Text ⟶ Numbers)**
  - **Purpose:** Convert text into searchable vectors
  - **Used in:** setup.py (PDFs) + tutor_agent.py (user queries)
  - **API Call:**
    ```
    POST https://api.euron.one/api/v1/euri/embeddings
    Headers: {"Authorization": "Bearer <api_key>"}
    Body: {
      "input": "Light travels in straight lines",
      "model": "text-embedding-3-small"
    }
    ```
  - **Response:**

```
{
  "data": [{
    "embedding": [0.123, -0.456, 0.789, ..., 0.234]  # 1536 numbers
  }]
}
```

- **When Used:**
  - ✅ **Setup Phase**: Convert PDF chunks to vectors
  - ✅ **Runtime Phase**: Convert user search queries to vectors

- **Endpoint 2: /chat/completions (Prompt → Human Text)**
  - **Purpose:** Generate human-readable responses
  - **Used in:** tutor_agent.py (_call_ai method)
  - **API Call:**

```
POST https://api.euron.one/api/v1/euri/chat/completions
Headers: {"Authorization": "Bearer <api_key>"}
Body: {
  "messages": [{"role": "user", "content": "Create a weekly study plan..."}],
  "model": "gpt-4.1-nano",
  "max_tokens": 2048,
  "temperature": 0.7
}
```

  - **Response:**

```
{
  "choices": [{
    "message": {
      "content": "# 🔬 Science Learning Roadmap..."  # Generated roadmap
    }
  }]
}
```

  - **When Used:**
    - ✅ **Roadmap Generation**: Create study plans
    - ✅ **Kid Chat**: Answer questions in kid-friendly way

🔄 API Integration Flow:

- **During Setup (setup.py):**

```
# For each PDF chunk:
text_chunk = "Light travels in straight lines..."
```

```
# Call embeddings API:
embedding = EuriaiEmbeddings().embed_documents([text_chunk])
# Result: [[0.123, -0.456, 0.789, ...]]

# Store in FAISS:
faiss_index.add_documents([{text: text_chunk, vector: embedding}])
```

- **During Runtime (tutor_agent.py):**

```
# Step 1: Convert user query to vector
query = "CBSE 10th Science topics"
query_vector = EuriaiEmbeddings().embed_query(query)
# Uses: /embeddings endpoint

# Step 2: Search similar vectors in FAISS
similar_docs = faiss_index.similarity_search(query_vector)

# Step 3: Generate response with context
prompt = f"Create roadmap based on: {similar_docs}"
response = requests.post("/chat/completions", json={"messages": [{"content":
prompt}]})
# Uses: /chat/completions endpoint
```

💰 API Usage Summary:

Operation	**Endpoint**	**Frequency**	**Purpose**
Setup PDFs	/embeddings	Once per PDF chunk	Convert text to vectors
User Query	/embeddings	Once per search	Convert query to vector
Generate Response	/chat/completions	Once per request	Create human text

🎯 Why Two Different Endpoints:

1. ***/embeddings***: Mathematical similarity search
    - **Input:** "photosynthesis"
    - **Output:** [0.123, -0.456, ...]

    - **Purpose:** Find related content
2. **/chat/completions**: Human communication
    - **Input:** "Create a study plan about photosynthesis"
    - **Output:** "# Week 1: Introduction to Photosynthesis..."
    - **Purpose:** Generate readable responses

🎯 6. AI Tutor v1.0.0 - Complete Core Summary

🏗️ Architecture Overview:

```
User Interaction (Gradio UI)
      ↓
  App Router (app.py)
      ↓
 AI Logic (tutor_agent.py)
      ↓
 Vector Search (FAISS) + API Calls (Euriai)
      ↓
  Response to User
```

📁 Core Files & Responsibilities:

- **app.py**: User interface + routing
- **tutor_agent.py**: AI logic + RAG implementation
- **setup.py**: PDF processing + vector index creation
- **euriai_embeddings.py**: Text-to-vector conversion wrapper
- **.env**: API key storage
- **data/syllabi/**: User's PDF files
- **data/vector_store/**: Searchable index (auto-generated)

🔄 Two-Phase System:

1. **Setup Phase** (One-time): PDF → Text Chunks → Vectors → FAISS Index
2. **Runtime Phase** (Per request): User Query → Vector Search → AI Generation → Response

🎛️ User Experience:

1. **Add PDFs** to data/syllabi/ (format: Board_Grade_Subject.pdf)
2. **Run Setup**: python setup.py (processes PDFs)
3. **Launch App**: python app.py (starts web interface)
4. **Select Options**: Grade, Board, Subject, Age
5. **Get Results**: Learning roadmaps or chat responses

🔌 **External Dependencies:**

- **Euriai /embeddings**: Text → Vector conversion
- **Euriai /chat/completions**: Prompt → Human text generation
- **FAISS**: Fast vector similarity search
- **Gradio**: Web UI framework
- **LangChain**: PDF processing + document handling

💡 **Core Innovation:**

- **RAG (Retrieval-Augmented Generation)** ensures AI responses are:
  - ✅ **Accurate**: Based on real syllabus content
  - ✅ **Relevant**: Filtered by exact board/grade/subject
  - ✅ **Age-Appropriate**: Customized prompting for kids
  - ✅ **Educational**: Structured learning plans

This is a complete, working AI tutor system that turns any PDF syllabus into an interactive, intelligent learning assistant! 🚀✨